Indian Institute of Technology Patna
CS1101- Foundations of Programming

Lab10: Gdb, Recursion and Dynamic Memory

Date: 18-10-2025

Work in a separate directory named **Lab10**.

Task1: GNU Debuger Tutorial

GDB is a standard debugging tool that comes pre-installed on all Linux platforms**.** We are introducing it now so that when you begin working on the upcoming C projects, you will have a reliable way to diagnose issues in your code. In this exercise, you'll be given two very simple C programs containing errors, and your task will be to use GDB to debug them. Becoming proficient with GDB is not only valuable for this course—it's also a practical skill you can highlight on your resume. Various examples are provided that follow the slides. In your lab record, document any new GDB commands you encounter, along with brief notes on your observations.

Task2: Carefully explore the provided *Recursion.c* program. Study each segment demonstrated in the code, and document in your lab record any constructs or features that are new to you, along with brief notes on your observations.

Task3: Carefully review the provided matrix addition program. Then, using dynamic memory allocation, implement a matrix multiplication algorithm that works for any M×N size.

In Record: Task1 , Task 2 and Task 3:

```c
#include <stdio.h>
#define N 10
void print_matrix(char *name, int M[N][N]) {
    printf("%s:\n", name);
    for (int i = 0; i < N; ++i) {
        for (int j = 0; j < N; ++j) {
            printf("%4d ", M[i][j]);  }
        printf("\n");
    }
    printf("\n"); }
int main(void) {
    int A[N][N];
    int B[N][N];
    int C[N][N];

    /* Initialize A and B, compute C = A + B */
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            A[i][j] = i + j;
            B[i][j] = i - j;
            C[i][j] = A[i][j] + B[i][j];
        }
    }
    /* Print matrices */
    print_matrix("Matrix A", A);
    print_matrix("Matrix B", B);
    print_matrix("Matrix C = A + B", C);
    return 0;
}
```

```c
#include <stdio.h>
#include <stdlib.h>
#define N 10
int main(void) {
    int *A = malloc(N * N * sizeof(int));
    int *B = malloc(N * N * sizeof(int));
    int *C = malloc(N * N * sizeof(int));
    if (!A || !B || !C) {
        printf("Memory allocation failed\n");
        return 1;
    }
    // Initialize and compute C = A + B
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            A[i * N + j] = i + j;
            B[i * N + j] = i - j;
            C[i * N + j] = A[i * N + j] + B[i * N + j];
        }
    }

    // Print A
    printf("Matrix A:\n");
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            printf("%4d ", A[i * N + j]);
        }
        printf("\n");
    }
```